

Architectural Design of Digital Integrated Circuits
Prof. Indranil Hatai
School of VLSI Technology
Indian Institute of Engineering Science and Technology, Shibpur, Howrah

Lecture – 32
Multiplier Architecture (Contd.)

Hello everyone. Welcome back to the course on Architectural Design of ICs. So, in the last class we have seen Booths algorithm then sign extension; that means, template method for reduction of the sign extension. So, after that what we have seen? Again we have seen that; that means, block diagram representation of Booths multiplier that we have seen ok.

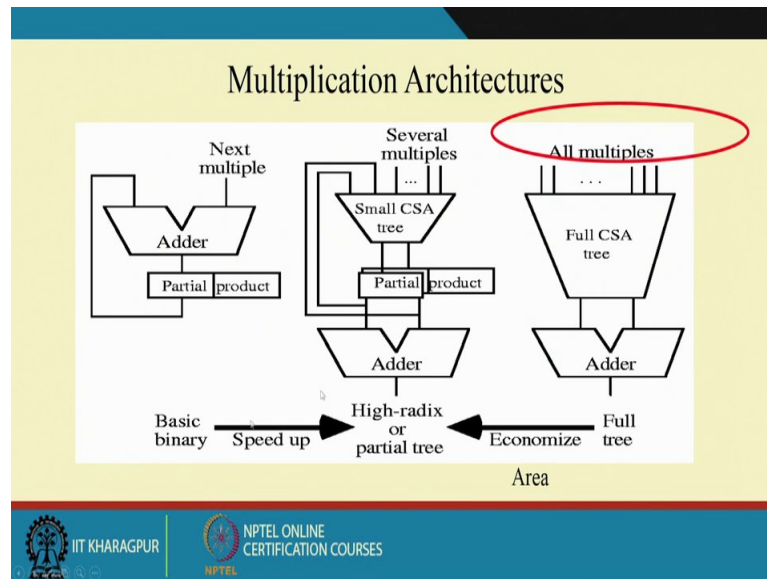
Now, we will see and whenever we are; that means, after Booths multiplication or after using of this Booths algorithm, we are getting the number basically we use Booths algorithm to reduce the number of partial product. So, once we generate the partial product then again I have to add those partial products to get the final results. So, those whenever we are doing those partial products adding unit so, at that time I can use this tree reduction method ok.

So, apart from that instead of Booths multiplication, there are tree multiplications too like this Wallace tree multiplication and another tree multiplications are there. So, there also we are doing this; that means, this reduction in the depth of the adder in chain operation using this binary tree.

So, in today's class we will see those tree multiplication or how we can reduce the corresponding number of delays or the number of full adders, in what aspects I can basically do the reduction that we will see in today's class.

So, let us start with this tree multiplication.

(Refer Slide Time: 01:47)



So, in this the first multiplier architecture, this is nothing, but the serial one so; that means, the ones it is coming using one adder only this will be run in repeatedly, and I will get the corresponding results ok. And if I use all the multiples together using carry save adder tree and finally, at the very last stage I need 1 carry propagate adder. So, and then I can get the results which is basically the full tree this is one method

And then there is another method, where I can consider partial tree; that means, the whole tree instead of this whole tree, if I consider only partial tree and I use or I calculate some of the those using carry save adder and I use very few parts of this partial products and then finally, I add those in a carry propagate adder so at that time also I can do that. So that means, the tree; that means, this multiplier are categorised into 3 different architectures.

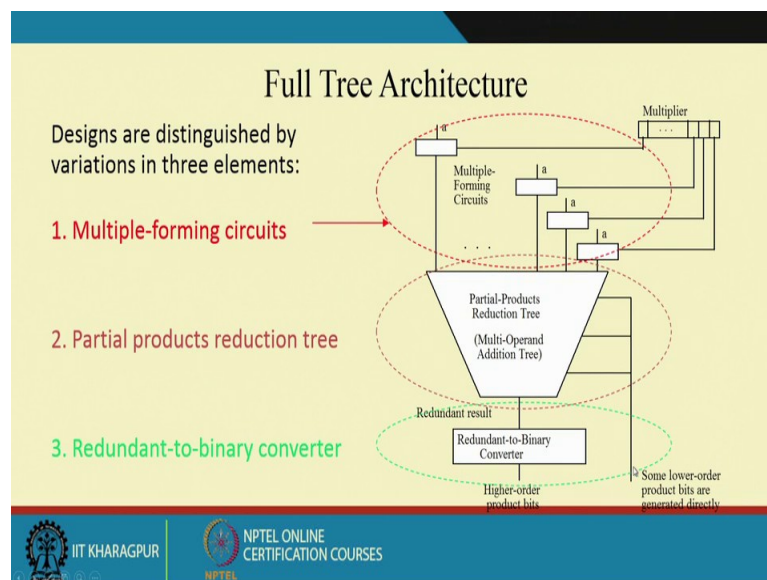
So, one is that where I am considering all the multiples one is that where I am considering only single multiples, and one architecture is that there where I am considering the partial multiples, so then if you use this particular things. So, at the time area wise there is better, but speed wise this is poor. At this case area wise this is poor, but speed wise this will be much more better speed means computation time. I am not telling that speed means the frequency of operating frequency of that particular circuit, but the corresponding that computation time for the multiplication, that is reduce in this particular case in case of instead of this.

But in case of; that means, some part in serial; that means, in this middle case what we

are doing? We are not doing here what in the first architecture what we are doing we are considering Single multiples and then we are serially doing and in the last architecture what we are doing? We are considering all the multiples and then parallelly we are doing, but in this mixed culture this mix architecture what we are doing? We are considering serial some of the things in serial and those serials are basically running in parallel ok.

So that means, mixture of this serial and parallel together gives me the benefit of area as well as the speed both ok. So, that is why we; that means, this mixed architecture for serial and parallel they are basically optimise in terms of power and area in with respect sorry power and speed or the; that means, the in the in terms of computation time, for in compared to both of this circuits ok.

(Refer Slide Time: 05:28)



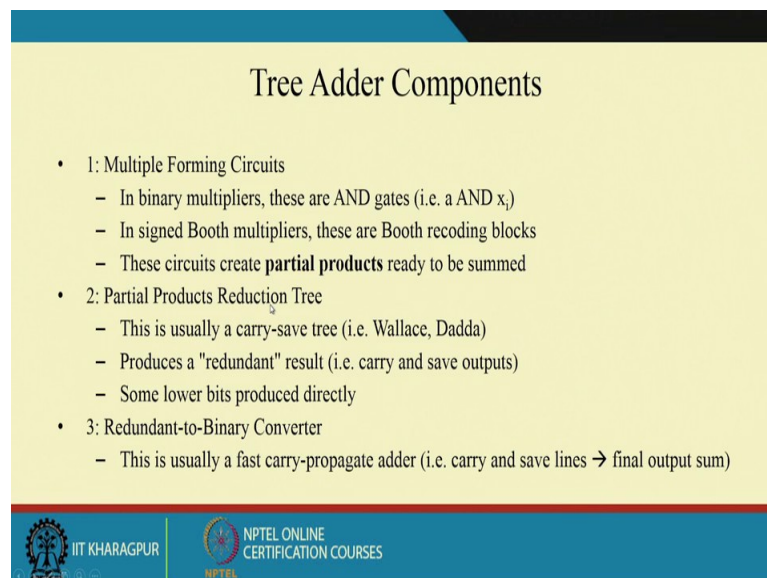
So, if I consider this full tree architecture. So, at that time in this tree multiplier architecture it has 3 part one part is this multiple forming circuit. So, multiple forming circuit is nothing but the generation of the partial products so; that means, this also can be named as this partial product generation unit. So, those are nothing, but the AND operation because each bit is ANDed with that means, each bit of the multiplier that is ANDed with each bit of the multiplicand so; that means, that is why this is; that means, that those are the multiples. So, that is why this particular circuit is named as multiple forming circuit.

Then the next part is this partial products reduction tree. So, after generating the partial

products, what I have to do? I have to basically add those partial products. So, while we will add those, at that time I can use this binary tree or this tree; that means, addition operation to reduce the corresponding delay for that and the third one is that, then redundant to binary converter. So, once I generate this then this is for; that means, the final output generation unit ok.

So, these part is basically the main part, where I am doing this partial product reduction tree. So, this basically; that means, this is the major research area or the focus area where people have tried so, many that means, way out to find out that this to find so, many way out to reduce the tree in such a way so, that the multiplier becomes high performance ok.

(Refer Slide Time: 07:40)



Tree Adder Components

- 1: Multiple Forming Circuits
 - In binary multipliers, these are AND gates (i.e. a AND x_i)
 - In signed Booth multipliers, these are Booth recoding blocks
 - These circuits create **partial products** ready to be summed
- 2: Partial Products Reduction Tree
 - This is usually a carry-save tree (i.e. Wallace, Dadda)
 - Produces a "redundant" result (i.e. carry and save outputs)
 - Some lower bits produced directly
- 3: Redundant-to-Binary Converter
 - This is usually a fast carry-propagate adder (i.e. carry and save lines → final output sum)

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, that means, these are the; that means, things which are related to this multiple; that means, this tree adder components or this tree adder multiplier architecture, I have already discussed that multiple forming is nothing, but the combinations or this. That means, it uses those AND gate for bitwise multiplication ok.

Then partial products tree that is basically uses the carry save adder tree, that is the examples are like Wallace tree and Dadda tree that we will see and produces a redundant results, that is carry and save outputs and some lower bits produce directly. So that means, that some of the things some of the bits are generated and the lower bits are directly it comes as the output. So, after getting though; that means, the after adding those finally, it will be of 2; that means, 2 number will be there. So, while I will get only


2 numbers. So, at the time I need to use 1 first adder either you can use carry propagate adder or conditional sum adder for those kind of addition. So, at that time this will basically comes to as the part of this redundant to binary converter.

(Refer Slide Time: 09:04)


1. Multiple Forming Circuits

Creates 5 partial products, each requires 5 AND gates → 25 AND gates


		a_4	a_3	a_2	a_1	a_0							
	x	x_4	x_3	x_2	x_1	x_0							
ax_0	2^0						a_4x_0	a_3x_0	a_2x_0	a_1x_0	a_0x_0		
ax_1	2^1						a_4x_1	a_3x_1	a_2x_1	a_1x_1	a_0x_1		
ax_2	2^2						a_4x_2	a_3x_2	a_2x_2	a_1x_2	a_0x_2		
ax_3	2^3						a_4x_3	a_3x_3	a_2x_3	a_1x_3	a_0x_3		
ax_4	2^4						a_4x_4	a_3x_4	a_2x_4	a_1x_4	a_0x_4		



IIT KHARAGPUR



NPTEL ONLINE
CERTIFICATION COURSES



NPTEL

So, if we chose one example of that. So, at the time it will be much more clear to you; if I have to multiply 5 bit 5 bit. So, I have to generate 5 partial product set ok. So, each set containing 5 this numbers. So that means, this 5 partial products and each requires 5 AND gate to produce this each of this multiple, so total I require 25 and gates in multiple forming circuit, this is nothing but the partial product generation unit.

(Refer Slide Time: 09:41)

2. Partial Products Tree Reduction (Wallace)

- After partial products created, must sum them together
- Wallace tree: reduce number operands as soon as possible, 4 x 4 example below

Wallace Tree: 2 carry-save levels, 5 FA, 3 HA, 4-bit CPA

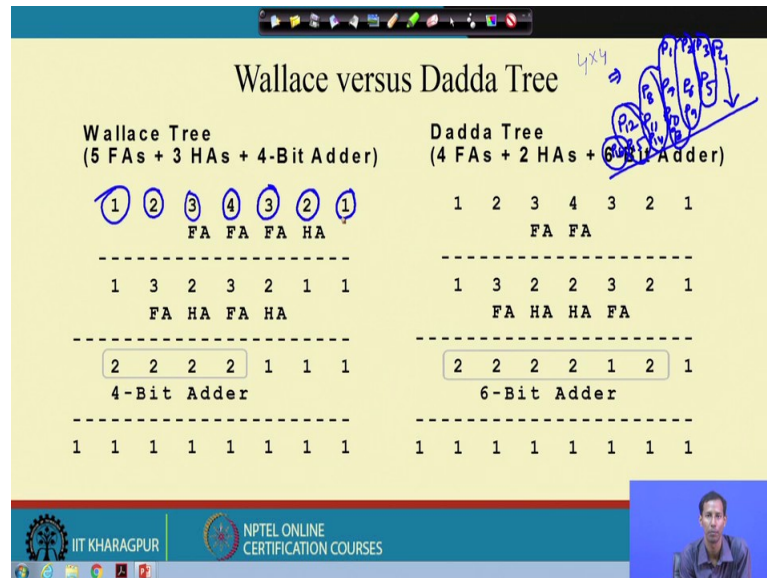
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, after that what I said that I can use different tree reduction method for this partial tree generation; that means our addition operation. So, in Wallace tree what we do in Wallace tree as soon as possible, this multiples are I will get those multiples I will try to add this ok. So, and while I will add after that at the very last stage I will get only 2 input of that ok. So, and finally, using a carry propagate adder at this level, I will get the final results.

So, here you see I need to add for 4 cross 4; that means, this case I have to use this 2 carry save level, which is 1 here 1 here and 2 set of 5 full adder cell as well as 3 half adder cell and 4 bit carry propagate adder at this is the carry propagate adder. And these are the; that means, half adder and full adder cell ok. So, total 18 number. So, 5 full adder among them 5 full adder and 3 half adder.

If I use this Dadda tree method, in Dadda tree method that is as late as possible we have to add. In Wallace tree as soon as possible we are adding, but in Dadda tree method as late as possible we are adding. So, there are also 2 carry save levels along with that I have to use 6 full adder 2 half adder and 6 bit carry propagate adder in the previous case how many I am using? 4 bit carry propagate adder and 8 adder cell among; that means, in between 5 full adder half adder, but here 6 adder cell among them full; that means, 4 full adder cell 2 half adder and instead of 4 bit in case of Wallace tree, 6 bit carry propagate adder I have to use in Dadda tree method.

(Refer Slide Time: 11:53)



So, then if i; that means, take 1 example of that so, at that time you can see; that means, what is this number 1 2 3 4; that means, 3 2 1 what is this numbers ok? These numbers are basically coming these numbers are basically coming for because what we have considered? We have considered this 4 cross 4 multiplier right. So, 4 cross 4 multiplier, it gives me what at the very first I will get P 1 P 2 P 3 P 4 sorry P 4 right then what I will get? I will get sorry P 5 P 6 P 7 P 8 something like this, then P 9 P 10 P 11 P 12 then P 13 this is 13 P 14 P 15 and P 16.

So, if I just want to add, add this particular position how many numbers? 1. So, this 1 corresponds to this then at this position how many numbers? So, 2; so this number corresponds to this then at this position how many? 3. So, this 3 corresponds to this then at this how many? 4. So, 4 corresponds to this, then this is 3, 3 over here, 2 2 over here and 1 1 over here ok. So, that is why this numbers are basically how many numbers I have to add at this particular position that indicates why this numbers.

So, now if i; that means, now whenever I will consider at the time for this 2 numbers if I have to add. So, at that time I need one half adder over here, for 3 I need one full adder for 4 one I need 1 full adder for 3 I need 1 full adder. So, after that this will produce the carry over at this fourth stage. So, this will; that means, forward the carry to the this stage and this will forward the carry to this stage.

So, after that; that means, at this stage so, this carry will be forwarded over here. So, 3 I

have already consumed, but it produce 1 and the carry produce from here; that means, from this particular level. So, that will become 2 over here. So, three I have consumed one I have left ok. So, 1 one has been generated from this there is another one and the carry from here. So, that number now it will become 3. So, 3 I have already consumed and produced one. So, one over here and one carry from this particular stage. So, this will be 2. So, here there was 2, but one carry from this particular level. So, this will become 3 now.

So, then after that I have to use 1 half adder for these 2 full adder for these 2 half adder for sorry full adder for these 3 2 for this half adder 3 for full adder. So, once again I consumed; that means this half adder for these. So, if this will produce one these 3 that one and carry from this. So, it requires 2. So, then again 2 it is consumed produce one; one from this particular carry. So, that is 2 3 consumed by full adder produce one output. So, carry from this particular level so, that becomes 2. So, initially there was 1 and the carry from this particular level that will effect to the next stage. So, that will become 2 so; that means, 1 1 1 that directly come to the first products, but for these 2 I need one 4 bit carry propagate adder ok. So, this is the Wallace tree method.

What happens in Dadda tree method? So, the same numbers I will get, but here you see I am choosing at the middle where I am getting this 4 and 3 ok. So, I have started from the maximum position, addition at the where I am getting the numbers are maximum. So, full adder for these 4 and full adder for these 4 at the very first level.

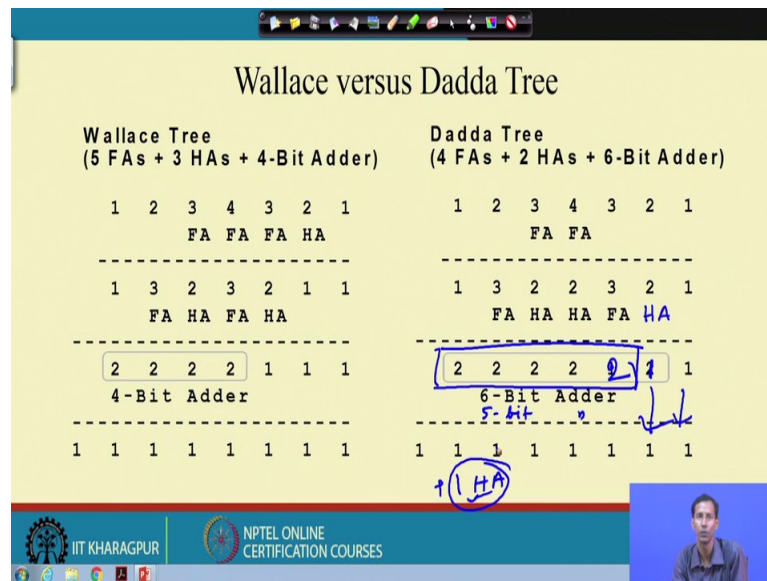
Then in the next so, 2 is already there, 3 is already there then here I have consumed 3 and there is already 1. So, and it produce 1 so; that means, sum along with the another input that is gives me 2; for this it produce 1 the carry from this particular level. So, 2 and the carry; that means, the generated from this stage, that will come over here so, that becomes now 2 becomes 3.

. So, in the next level what I will do? I will use for 3, I will use full adder for these two I will use one half adder, these two I will use one half adder, these 3 I will use full adder. So, at this particular stage; so 2 is 2 3 will produce only 1. So, then 2 produce 1 and the carry from this particular level so that will produce 2, so 2 produce 1 carry from the other level 2 sorry then 3 consumed and produce 1 carry from the previous level. So, that will produce 2. So, 1 here the carry propagated from; that means, the previous stage that

becomes 2.

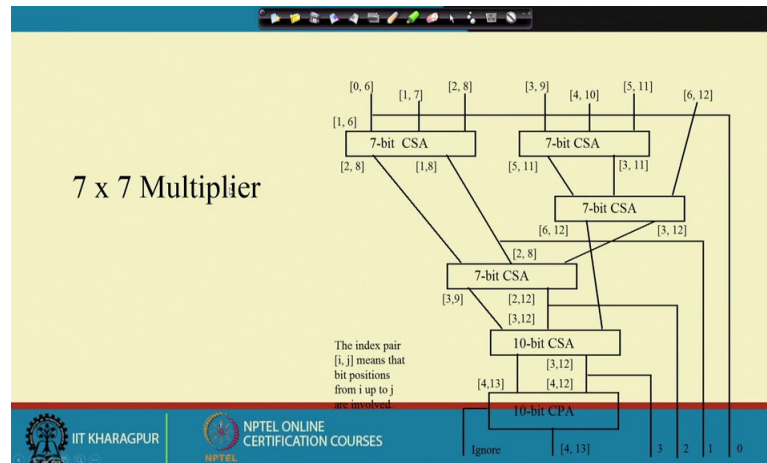
So, now, this 2 2 2 2 1 and 2 that I need to use and that will be using 6 bit carry propagate adder, now I will get the corresponding digits. So, here apart from that another thing also here we are going to; so at this particular level whenever I am considering. So, I can use 1 half adder over here. So, if I use half adder so, at that time if I use one half adder over here so, at that time what will become?

(Refer Slide Time: 18:44)



It will become 1, this will become 2 at that case so; that means, these 2 will directly come and I can use this at the time what I require? I will require 5 bit adder for doing this, but at what cost? I have increased by the hardware by one half; that means, here what I am doing? I am putting 1 extra half adder, but I can reduce the adder 3 by this 6 bit to 5 bit that also I can do. So, this is the basic difference between Wallace tree method and Dadda tree method

(Refer Slide Time: 19:42)



So, this is the; that means, corresponding block diagram level representation or block diagram level, this implementation technique of 7 cross 7 multiplier this using this tree reduction method ok. So, whether that is Wallace tree or by that Dadda tree method. So, this is at the very final level I have to use this carry propagate adder to find out the final results. And this is nothing but whatever that in the previous example whatever we have seen? This is nothing but the block level representation of that.

(Refer Slide Time: 20:23)

Different Reduction Architectures

- Tree architectures (Wallace, Dadda) are quite irregular and can cause VLSI implementation issues
- Seek other methods for partial product reduction which have better VLSI properties
- Two methods:
 - 11:2 compressor (counter) method occupies narrow vertical slice
 - Create one column, replicate side-by-side
 - A carry created at level i enters level $i+1$
 - Balanced delay tree
 - 11 inputs, 2 outputs
 - 4:2 binary tree

So, then in; that means, tree reduction method what I; that means, say the what I find out that, the corresponding that methods are very much irregular in the sense. So, somewhere I am using full adder, then I am half adder, there is the structure is not that much regular

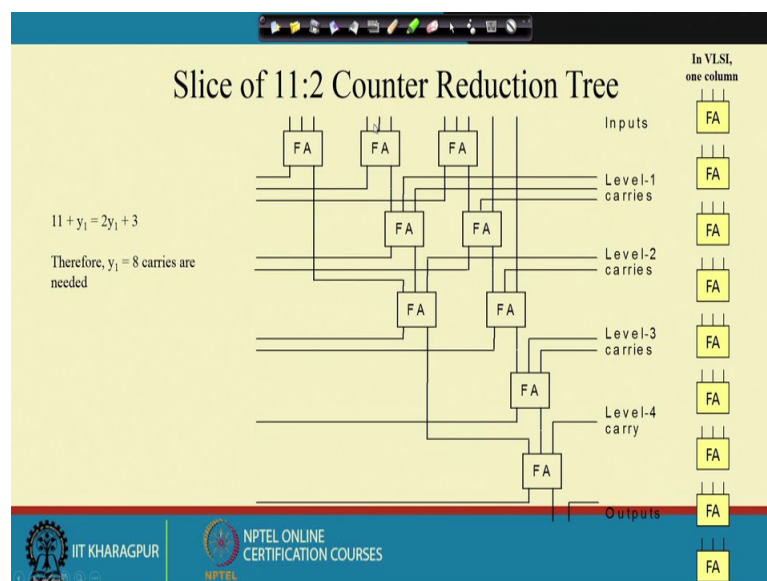
ok. So, that is why this vlsi implementation of this particular this Wallace tree or Dadda tree, that can create some of the issues.

So, to remove or to get out of those problems what we can use? You we can use this 11 is to 2 compressor or we can use this 4 is to 2 compressor. So, these particular methods basically are very much useful as this structure is regular. So, this is that means for the VLSI implementation of those multipliers architecture that give some of the mileage or give some of the advantage to the circuit realisation.

So, in 11 is to 2 compressor the method basically applies in the vertical direction and it creates one column replicates side by side and a carry created at level i enters into level i plus 1, the three methods are very much balanced it consumes 11 inputs together and produce 2 outputs finally so that means, the thing is that; that means, suppose in one case if I am having the partial products up to P 11 that means the numbers are something like this I need to add for this one particular position. So, at the time I can use this 11 is to 2 compressor to add this terms ok.

So, the same thing if I used 4 is to 2 binary. So, at that time this will be only P 1 to P 4 that will also produce 2 binary one; that means, sorry 2 output 1 is sum another 1 is carry. So that means, whatever is the corresponding the; that means, the numbers or the sum that will be produced through this 11 is to 2 compressor or 4 is to 2 compressor.

(Refer Slide Time: 23:07)



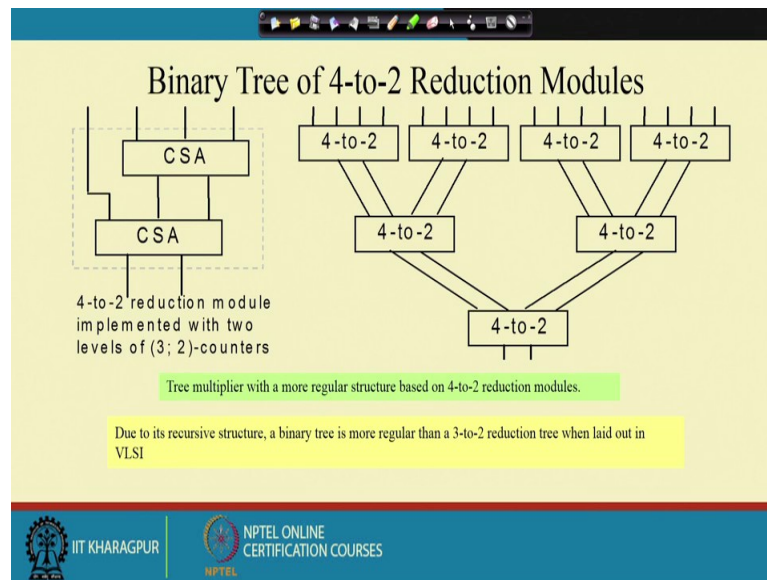
So, if I just see the corresponding architecture; that means, the inside of this 11 is to 2 compressor. So, at that time you see that the sales are like VLSI implementation that has coming like this. So, here initially what is there; that means, I have at this very first level if this is the; that means, level 0. So, then 3 full adder cell, I have used where it is consuming total 9 numbers of inputs and it produced what? 3 numbers of carries and 3 numbers of sums; so in the next level.

So, in the previous case what I says? Level, I carry that enters into level i plus 1 number 1; that means, level. So, here whatever that sum I am getting from this, I have used 2 full adder cell one sum I have used here and one sum I have used in this full adder cell and I have produce how many carry? So, total 3 carry. So, two carry comes for this particular full adder cell one carry comes at this particular case.

So, then again at level 1 I am producing 2 output carry and 2 sum then again that 2 sum again enter into 2 full adder cell and the 2 carry are basically coming to the 2 carries are coming to the corresponding 2 full adder cell. Then another of this sum is already left. So, that is coming to this full adder cell and another position to this full adder cell is already; that means, left. So, another of the input which is coming over here and in this particular case also another input is coming over here; so that means, at level 2 at level 2 again I am doing using 2 full adder cell, I am producing 2 carries and 2 sum. So, then this sum is basically coming over here and 2 2 carry 2 carry from the previous level. So, those 2 caries are coming in this stage

So, then again this sum is coming, this sum is coming and the carry from this particular full adder the adder coming over here and I am getting final sum sorry final sum and final carry at this particular adder tree. So, this structure is very much regular, I have not used any like here full adder then half adder then carry propagate; that means, at the very final level I have to use this carry propagate adder that is the different is. But unlike this Wallace tree and Dadda tree method. So, if I use this for the each of this position; so if I use this kind of structure or this compressor method. So, this is giving me one very much regular structure, which is very much useful in VLSI architecture.

(Refer Slide Time: 26:22)

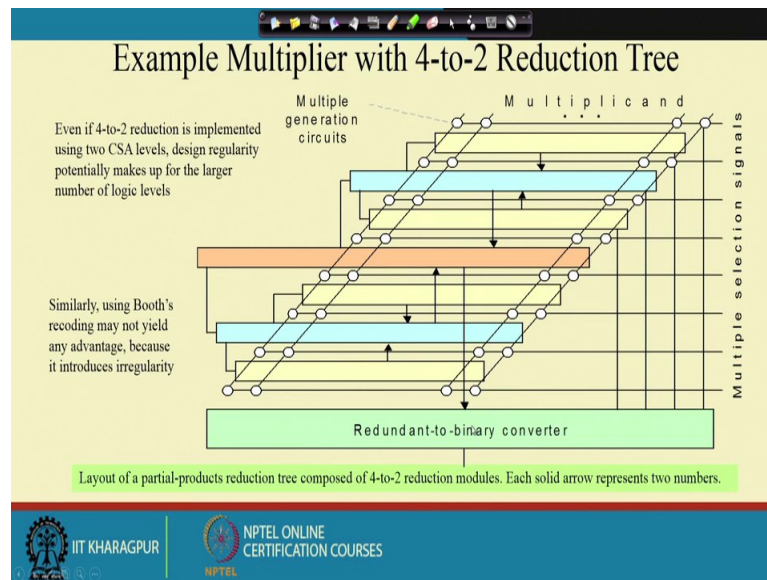


So, for; that means, 4 is to 2 reduction method. So, it will at the very first it will consider one carry save adder tree; that means, one carry save adder level then again that sum and carry along with the another carry that will produce 2 output. So, now, for 16 I will use this 4 of them, then again 4 will be generated from this then again that will be. So, using this 3 level now can now I can add 16 number of multiples using this 4 is to 2 compressor, and you see this structure is very much regular in nature ok.

So, that is why this tree multiplier which is more regular structure based on this 4 is to 2 this reduction modules and due to its recursive structure a binary tree is more regular than a 3 to 2 reduction tree when laid out in VLSI. So, whatever we are doing in; that means, really that in Wallace tree and Dadda tree, that are either 2 is to 2 or 3 is to 2; that means, this carry save adder we are using.

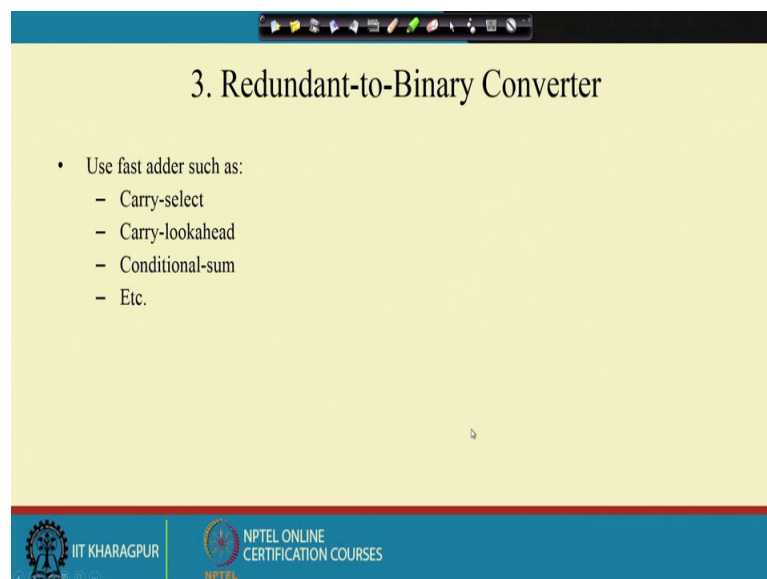
So, 3 is to 2 means we are consuming 3 and we are producing 2 outputs basically that is the full adder cell in half adder cell that is 2 is to 2 means 2 input it is taking and 2 output it is producing. But here what we are doing? We are using inside of that we are using 3, but whenever we are considering the multiple side at that time combination of 4 we are considering and this gives me a regular structure.

(Refer Slide Time: 28:11)



So, this is the; that means, the layout of the corresponding partial product reduction tree composed by this 4 is to 2 reduction module ok. So, as I said that this gives me a better; that means, or a regular structure while we are implementing it in VLSI.

(Refer Slide Time: 28:39)



So, after that what I have to use, what I left; the third which is redundant to binary convertor. So, I can use any of the fast adder which are any of this adder; that means, which will be this any of this adder, where that will be carry select adder or carry look ahead adder or carry propagate adder or conditional sum adder any of this adder for the

final 4 bit or 6 bit or 8 bit addition operation while this terms comes to 2. So, at that time; that means, after a tree reduction the final terms will come whenever that will reduce; that means, come to the 2. So, at the time you have to use any of this fast adder to get the better result or to reduce the delay for that particular multiplier design what you are intended to do.

(Refer Slide Time: 29:31)

Wallace Tree multipliers

- Use the 3:2 counters and 2:2 counters
- Number of levels of $= \log(32/2) / \log(3/2) \approx 8$
- Irregular structure
- Fast

Input: Sum
Output: Carry

2:2 counter 3:2 counter

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

62

So, this is the; that means, in Wallace tree the number of levels you can get by log of this.

(Refer Slide Time: 29:41)

Wallace Tree and Ripple Carry Adder Structure. Of 8*8 multiplier With Pipeline

Partial Product PP0,PP1,PP2(15 downto 0)

Partial Product PP3(15 downto 0)

Ripple Carry Adder

Pipeline Register

Critical Path

P16 P15 P14 P13 P12 P11 P10 P9 P8 P7 P6 P5 P4 P3 P2 P1 P0

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

63

So, this I think I have already discussed. So, this is the using carry ripple adder also you can do. So, we have we have used mostly we have used what for carry save adder and we use carry propagate adder while we are considering this tree reduction method, but ripple carry adder also you can use, but if you use ripple carry adder. So, at the time you will get the delay will be more so that means, to reduce to improve the corresponding delay or the; that means, the speed of operation. So, that for we have to use this carry save adder or you have to use this carry propagate adder.

(Refer Slide Time: 30:14)

	Array Multiplier	Modified Booth Multiplier	Wallace-Tree Multiplier	Modified Booth Wallace Tree Multiplier	Twin Pipe Serial-Parallel Multiplier	Behavioral Multiplier
Area - Total CLB's (#)	3076.50	2649.50	3325.50	2672.50	490.00	2993.50
Maximum Delay D(ns)	35.78	24.43	18.93	18.53	107.52 (3.36x32)	49.33
Total Dynamic Power P (W)	7.52	6.33	7.46	6.41	0.28	6.24
Delay * Power Product (DP) (ns W)	268.98	154.64	141.14	118.76	30.62	307.58
Area * Power Product (AP) (# W)	23128.20	16771.60	24793.93	17127.79	139.54	18665.07
Area * Delay Product (AD) (# ns)	1.10E+05	6.47E+04	6.30E+04	4.95E+04	5.27E+04	1.48E+05
Area * Delay ² Product (AD ²) (# ns ²)	3.94E+06	1.58E+06	1.19E+06	9.18E+05	5.66E+06	7.28E+06

So, this is the comparison tables of all the multipliers we have discussed till now. So, we have used or we have; that means, seen Array multiplier, we have seen this Booth multiplier we have seen this Wallace tree multiplier or else we can use this hybrid of this; that means, Booth algorithm and then Wallace tree multiplier, and we have seen this serial parallel multiplier. And this is behavioural multiplier means that nothing but the generic multiplier architecture that is that is the shift and add based multiplier. And this is basically the PGA implementation and along with that it shows that in serial parallel it consume the more. That means, the lesser area in compared to all of the architecture ok; that means, we are generating the in a serial fashion we are using. So, that is why the time required and the computation time requirement is also more, if you see this table for that particular case

The then the next is that the Wallace that means, the modified Booth and Wallace tree

multiplier that you; that means, requires 2672 and the Booth multiplier requires 2649 whereas; that means, the lowest possible; that means, if I consider this parallel architecture, this gives me the lowest possible the number of area requirement, but here the delay requirement is 24.43, but the delay requirement for this is 18.53.

Here what we are doing? In Booth multiplier the area requirement is lesser, in Wallace tree this is the first addition operation; that means, the delay is reduced in compared to the Booths. So, while we are hybridizing these 2 architecture; that means, I will get the benefit of area as well as I will get the benefit of delay which I am getting in case of hybrid multiplier which is Booth plus Wallace tree. So, 2672 just minimal increase from this 2649 number and here 18.93 and here it is 18.53; that means, just a minimal increase in the delay too. So that means, I am getting 1 optimize circuit if I use hybrid architecture in this case.

So, rest are the delay power; that means, area power the power wise this circuit is also very good in compared to this, time requirement is much more higher on this and behaviour multiplier it; that means, consumes also a; that means, very large number of area as well as power as well as delay.

(Refer Slide Time: 32:56)

	Array Multiplier	Modified Booth Multiplier	Wallace-Tree Multiplier	Modified Booth-Wallace Tree Multiplier	Twin Pipe Serial-Parallel Multiplier	Behavioral Multiplier
Area	Medium	Small	Large	Small	Smallest	Medium
Critical Delay	Medium	Fast	Very Fast	Fastest	Very Large	Large
Power Consumption	Large	Medium	Large	Medium	Smallest	Medium
Complexity	Simple	Complex	More Complex	More Complex	Simple	Simplest
Implement	Easy	Medium	Difficut	Difficut	Easy	Easiest

So, in that means, overall overview if I say that area wise this modified Booth and Wallace multiplier, that are that are there is small and delay wise this is the fastest, power consumption wise the smallest is the parallel pipe, but as delay is very much

higher; that means higher side. So, this is the on the power side also we are getting much more savings in this hybrid architecture.

So that means, still now we have seen this unsigned multiplication. So, this is the; that means, this end of the unsigned multiplication, from the next class onwards we will signed multiplication techniques. It is the, this is not the end of this unsigned multiplication architectures, the as I said that there are different several research; that means, literatures are available on the internet or that is available in the Google or IEEE or any other scientific sites. So, if you are interested to design one. That means, if you are focusing on to design one high performance multiplier architecture, then please follow those literatures and you can also discuss via discussion forum.

Thank you for today's class.